Postel

E. Harslem
J. Heafner
RAND

Network Working Group
Request for Comments: 39

25 March 1970

COMMENTS ON PROTOCOL RE: NWG/RFC #36

We offer the following suggestions to be considered as additions to the April 28th 1970 protocol grammar specifications.

ERROR MESSAGES

<ERR> <Code> <Command in error>

It is desirable to include debugging aids in the initial protocol for checking out Network Control Programs, etc.

There are three classes of errors--content errors, status errors, and resource allocation or exhaustion. <Code> specifies the class and the offending member of the class. The command is returned to the sending NCP for identification and analysis.

Examples of status errors are: messages sent over blocked links and attempts to unblock an unblocked link. Examples of content errors are: an invalid RFC complete; a message sent on a link not connected; closing of an unconnected link; and an attempt to unblock an unconnected link. Examples of resource errors are: a request for a non-existent program and connection table overflow, etc. Resource errors should be followed by a <CLS> in response to the <RFC>.

QUERIES

<QRY> <My Socket> < >

or <QRY> <Your Socket> <Text>

Queries provide an extension to the <ERR> facility as well as limited error recovery, thus avoiding re-initialization of an NCP.

The first command requests the remote NCP to supply the status of all connections to the user specified by the user number in <My Socket>. The second is the reply; <Text> contains the connection status information. If an NCP wants the status of all connections to a remote HOST, the <My Socket> is zero.

PROGRAM TERMINATION NOTIFICATION

<TER> <My Socket>

This command supplements rather than replaces <CLS>. It severs all communication between a program and those programs in a given HOST to which it is connected. This command performs what would otherwise be handled by multiple <CLS> commands. <My Socket> contains the sender's user number.

HOST STATUS

<HCU>
<HGD>

These messages (HOST coming up and HOST voluntarily going down) are compatible with asynchronous, interrupt-driven programs, as opposed to the more conventional post/poll method.

TRANSMIT AND BROADCAST

<TRN> <Body>
<BDC> <Body>

Unlike the previous commands, these are not sent over the control link, but rather over links assigned to user programs. The prefix of <TRN> or <BDC> indicates, to the receiving NCP, the disposition of the message body. <TRN> indicates a message to be passed to a single process. <BDC> specifies to the destination NCP that the message is to be distributed over all receiving connections linked to the sender. In response to a system call by the user to an NCP requesting <BDC>, the NCP generates one <BDC> to each HOST to which the sender is connected.

RFC AND DYNAMIC RECONNECTION

This protocol is complex; it proliferates control messages; it causes queues (to become associated with re-entrant procedures) that are artificially imposed via the protocol (remote AEN assignment); and discounts the situation where only controlling process "A" has knowledge that slave process "B" should be "rung out" in a dynamic reconnection.

The <ERR>, etc., are suggestions for inclusion as additions in the April 28th protocol specifications. The above criticism is, of course, not intended to affect modification of the RFC structure by April 28th, nor to reflect on those who planned it. We have not studied the problem. It is meant, however, to voice our concern about complexity and resulting response times. This is a difficult problem and it deserves more study after we have exercised the current RFC specifications. We hope to offer constructive suggestions with respect to the RFC in the future.

JFH:hs